



(11) Publication number : **0 658 841 A2**

(12)

EUROPEAN PATENT APPLICATION

(21) Application number : **94480106.7**

(51) Int. Cl.⁶ : **G06F 9/46**

(22) Date of filing : **26.10.94**

(30) Priority : **16.12.93 US 168616**

(43) Date of publication of application :
21.06.95 Bulletin 95/25

(84) Designated Contracting States :
DE FR GB

(71) Applicant : **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

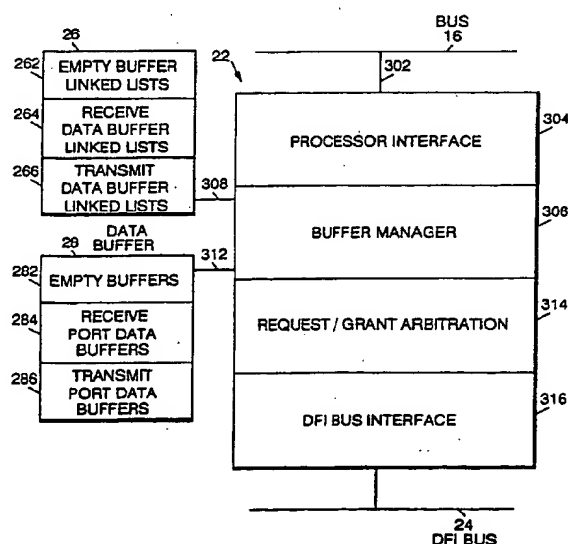
(72) Inventor : **Bass, Brian Mitchell**
4021 Old Sturbridge Drive
Apex, North Carolina (US)
Inventor : **Ku, Edward Hau-Chun**
115 Lochwood West
Cary, North Carolina (US)
Inventor : **Lin, Bou-Chung**
8712 Fort Macon Court
Raleigh, North Carolina (US)
Inventor : **Sanaye, Simin Hosne**
10301 Roadstead Way
Raleigh, North Carolina (US)

(74) Representative : **Lattard, Nicole**
Compagnie IBM France
Département de Propriété Intellectuelle
F-06610 La Gaude (FR)

(54) **A data processing system having a dynamic priority task scheduler.**

(57) A data processing system includes a server which has a task scheduler for receiving task requests for access, and granting access to system resources based upon a multidimensional scheduling technique wherein groups of tasks are assigned a priority level within a priority scheme, and within each group round robin scheduling assures each task will have access to system resources in turn and a dynamically programmable technique for responding to requests from time- dependent isochronous requests for system resources.

FIG. 3



stored in section 264 of linked lists 26. The data flow interface unit 30 then causes the received data to be stored in the receive port data buffers 284 assigned by buffer manager 306. When the data frame has been completely stored in receive port data buffers 284, LAN DFI 30 identifies a target port to buffer manager 306. Buffer manager 306 then transfers the data buffer linked list associated with the received data to a transmitted data buffer linked list which is stored in 266, and a notice is sent to the target port of the arrival of the data in data buffer 28 by buffer manager 306. After the data has been completely transferred to the target port, the target port releases the transmit data buffer 286, and buffer manager 306 places the pointers associated with the released buffers to the empty buffer linked lists 262.

Request/grant arbitration unit 314 responds to and grants requests based upon the arbitration strategy in accordance with the present invention. Data flow interface (DFI) bus interface 316 communicates between task scheduler 22 and data flow interface bus 24.

Referring now to FIGURE 4, a representative data flow interface unit such as LAN DFI unit 30 will be described. DFI unit 30 includes a data flow interface bus interface 402 which provides communication between data flow interface 24, received buffer 404, transmit buffer 406, and device drivers 408. Device drivers 408 communicate with the network such as network 106 shown in FIGURE 1 above. Interface 402, buffers 404 and 406 and device drivers 408 operate to transmit data between the external network 106 and data flow interface bus 24.

A request for access to system resources from another device such as device 108 connected to network 106 is presented as an input to request bus interface 412 which communicates the request through DFI bus 24 to request/grant arbitration unit 314 in task scheduler 22 (see FIGURE 3).

A typical data flow for system 100 is as follows:

Data arrives on line 36 at LAN DFI unit 30. DFI unit 30 issues a request for a free buffer to task scheduler 22. Task scheduler 22 issues an address pointer to a free buffer to LAN DFI unit 30. The data on line 36 is then moved into the buffer by LAN DFI unit 30 through task scheduler 22 to data buffer 28. If more space is required, LAN DFI unit 30 issues another request to task scheduler 22 for free buffer space. At the end of the data string input on line 36, LAN DFI unit 30 issues a command to task scheduler 22 to enqueue the data packet to a target data flow interface unit such as another LAN DFI, a WAN DFI such as 32, the data stored unit DFI unit 34, or processor 12. Task scheduler 22 places the data packet into the target DFI units queue which is stored in link list 26. Task scheduler 22 then raises a flag to the target DFI unit such as WAN DFI 32 to indicate a change in the queue status for the target DFI unit. WAN DFI

32 (the target DFI unit) reads the updated queue status. The target DFI (WAN DFI 32) issues a command to task scheduler 22 to obtain packet data, and task scheduler 22 returns information to WAN DFI 32 indicating data packet location and data length.

The target DFI 32 then moves the data packet from the data buffer 28 into its local buffer such as transmit buffer 406 (see FIGURE 4) for further processing. When the movement of the data packet is completed, the target DFI 32 raises a flag to task scheduler 22 indicating that the data move is completed. Task scheduler 22 then frees the data buffer 28 used for the operation for use on a subsequent data transfer operation.

OPERATION OF THE PREFERRED EMBODIMENT OF THE INVENTION

When more than one request for access to system resources is presented to task scheduler 22, tasks are granted access in accordance with a predetermined overall priority in which programmable time requests (isochronous time dependent requests) are granted highest priority, group 1 requests are next followed by group 2 requests, group 3 requests, etc. to the last or lowest priority which is group N requests where N is the highest group number requesting service at any time in the system. If a time dependent task is requesting use of system resources and its programmable time request is also active, it will be granted access. If the programmable time request is active, but the task is not requesting use of resources, then task scheduler 22 will give the grant of access to the highest priority group requesting access to system resources. The task within such highest priority group whose turn is active according to a round robin scheme and is also requesting access to system resources will receive the grant of access.

Referring now to FIGURE 5, the process for handling group requests will be described.

Each group of tasks is assigned a priority from highest to lowest. For example, group 1 may be assigned a highest priority and group N may be assigned a lowest priority. If, for example, a task from group 1 [task X(k)] is requesting access, a request active line is raised to the group section 502 of task scheduler 22.

If group 1 is active, task X(k) is subgranted access before any task of a lower priority group.

If there is more than one task from a particular group which is requesting access to system resources at any time, the group round robin circuit 602 of task scheduler 22 resolves the conflict.

Referring now to FIGURE 6, if a plurality of tasks all from a single group such as task X(j), task X(k), etc. are all requesting access to system resources, group round robin circuit 602 controls the granting of access to tasks in the same group in a manner similar to a

traffic control signal. Group round robin circuit 602 stores the task number of the last task in the group to have been granted access and updates that storage as each next task in the group is granted access in turn. Thus, one by one, all tasks in the group are granted access in turn. An exception might occur if a higher priority group or a time dependent isochronous task requests access to system resources.

Referring now to FIGURE 7 and to FIGURE 12, the programmable time scheduling circuit 700 of task scheduler 22 will be described for handling isochronous or time dependent task requests. Each task requiring isochronous support will have its task number written in task sequence random access memory (RAM) 702. The order of tasks in RAM 702 will determine the order of each task's time period. If certain tasks require support more often than others, the task number can be written into RAM 702 more than once. (Note task A appears four times in the example shown in FIGURE 7.) As address counter 704 increments through all addresses in RAM 702, register 706 is updated with the last value read from RAM 702. Register 706 stores a task number. The value in register 706 determines which task is given a particular time period. If that task is also requesting use of system resources, that task will receive the final grant from task scheduler 22. If the task is not requesting use of system resources during its allocated time period, the final grant will be based on group priority requests as described above with respect to FIGURES 5 and 6. As the programmable clock generator circuit 708 which is loaded with an initial value from processor 12 completes a cycle, address counter 704 increments and points to the next location in RAM 702. The value in register 706 will be updated, and another task number will be given the particular time slot. Since the programmable counter 708 is programmable under the control of processor 12, the time period length for each task can be updated as needed by processor 12. The count value initially stored in programmable counter 708 will determine the length of time that any task can have a highest priority time slot and how often the time window will change. Address bus 712 and data bus 16 allows processor 12 to initialize RAM 702 with the desired task sequence and allows processor 12 to store a desired time period value in programmable counter 708. Using a base clock signal as an input, programmable counter 708 generates an increment/read pulse once each time period. The counter will be reloaded with the original value stored in programmable counter 708 and continue counting. The increment/read pulse will go to rollover counter 704 (which is the address counter for RAM 702) and increment rollover counter 704 to a next value. Rollover counter 704 may also be programmable to allow the address range of RAM 702 to be altered as needed by processor 12. Rollover counter 704 is used as RAM 702 address during normal operation. The increment/read pulse

will also read RAM 702 and store the value in register 706. The output of register 706 provides an input to the final grant portion of task scheduler 22 as shown in FIGURE 8.

Referring now to FIGURE 8, the final grant section 802 of task scheduler 22 will be described. As final grant section 802 receives inputs from programmable time section 700 and group section 500, a first determination is made as to whether any task is currently active. If so, then such task is given final grant of access to system resources. If a time dependent task is not active, a determination is made if any group request is active. If a group request is active, then final grant of access to system resources is granted to the subgrant request task number from group priority circuit 502.

Referring now to FIGURE 9, a circuit for determining priority level among requests from different groups will be described. A number of active requests from groups 1, M, and N are presented to prioritization circuit 802. (Circuits for establishing a priority level among a number of inputs are well-known in the art and will not be described in detail in this application.) The highest priority active request will provide an output to multiplexer 804. As an example, assume that there is an active request from group M and an active request from group N and that there is no active request from any group having a higher priority than group M where the highest priority is group 1 and the lowest priority is group N. Prioritization circuit 802 will provide two outputs. A first output indicates that there is an active request (see FIGURE 5), and a second output to multiplexer 804 identifies the highest priority group (in this case group M) having an active request. Multiplexer 804 then selects the task associated with the highest priority group (group M) and provides an output (Task Grant) with an associated task identification number.

Referring now to FIGURE 10, and to FIGURE 6, a logic implementation of a round robin selection circuit will be described. Each request (REQ 1, REQ K, REQ N) is connected as one input to an AND gate which allows each request to set the associated latch 1002, 1004, 1006, one time for each cycle such that if, for example, request 1 should become active a second time before the round robin cycle has been completed, request 1 would be inhibited by latch 1008 from setting latch 1002. Thus, as each request REQ 1, REQ K, ... REQ N becomes active in turn, it is allowed to set its associated request latch 1002, 1004, 1006 once for each round robin cycle. Select logic made up of AND and OR gates 1010, 1012, 1014, 1016, 1018, and 1020 allow a particular request such as request 1 to be granted access to system resources only if its associated latch such as latch 1002 is set indicating that request 1 is active during its turn in the round robin scheme.

Referring now to FIGURE 11, select logic 1102

matches a task number having a predetermined time period for access to system resources as was described above with reference to FIGURE 7 with its associated active request (such as K REQ).

Only if the request is active during its assigned period will the output from select logic 1102 take priority in priority logic 1104 over the priority assigned to groups of non-time-dependent tasks. The isochronous tasks always have the highest priority in priority circuit 1104, but claim that priority only if there is a match between task number and request in select logic 1102. If there is no match output from select logic 1102, then the group request having the highest priority will be granted access to system resources.

Referring again to FIGURE 12, all of the task scheduling mechanisms described above are shown in a single diagram with the programmable task scheduling circuit near the top of the diagram and a group requests and round robin circuits towards the bottom of the diagram.

Claims

1. A data processing system having a dynamic priority task scheduler, comprising:
a processor for processing tasks;
a memory, for storing control information and data, associated with said processor;
a system bus for communicating control information and data between said processor and said memory;
a bus controller connected to said processor and said memory by said system bus and to a data bus for controlling data flow between said processor, said memory and said data bus, and for scheduling tasks presenting requests for access to said processor, to said memory or to other system resources, based on a task arbitration strategy which includes an integrated plurality of access granting means for handling isochronous as well as priority level based requests;
a data bus for communicating data from said processor or said memory to one or more data interface units;
one or more data interface units for communicating control and data signals between said data bus and a plurality of devices requesting access to said processor or said memory; and
a plurality of devices requesting access to said system resources.
2. A data processing system, according to claim 1, wherein said integrated plurality of access granting means further comprises:
first means for handling isochronous task requests having a dynamically variable access time requirement;

second means for handling task requests based on a predetermined priority of groups of requesters; and
third means for handling task requests from one or more tasks in a group having a same predetermined priority.

3. A data processing system, according to claim 1, wherein said integrated plurality of access granting means grants a highest priority to isochronous tasks requests.
4. A data processing system, according to claim 2, wherein said third means grants access to requests having a same priority level on a round robin basis.
5. A data processing system, according to claim 1, wherein said bus controller further comprises:
means for managing a plurality of data buffer registers in response to requests for access to system resources.
6. A method for dynamically scheduling a plurality of tasks in a data processing system, comprising the steps of:
requesting, by a plurality of tasks, access to elements of said data processing system;
determining a first priority level for each request presented;
granting access to a request having a highest first priority level for requests having different first priority levels;
determining if two or more requests for access have a same first priority level;
storing said two or more requests in a queue if said determining step indicates two or more requests have a same first priority level;
granting access to each of said two or more requests in turn such that all of said two or more requests are granted access to elements of said data processing system; and servicing said requests.
7. A method, according to claim 6, further comprising the steps of:
determining if any of said requests for access is for an isochronous task;
assigning a predetermined time period for each said isochronous task; and
granting a highest first priority level to any said request for an isochronous task during said predetermined time period assigned to said task.
8. A task scheduler for granting access to system resources in a data processing system in response to a plurality of requests for access, comprising:
means for determining a first priority level for

each of said requests for access;
means for granting access to a request having a
highest first priority level among two or more re-
quests having different first priority levels;
means for storing said two or more requests in a
queue if it is determined that two or more re-
quests have a same first priority level; and
means for granting access to each of said two or
more requests having a same first priority level in
a predetermined sequence such that all of said
requests are granted access to said system re-
sources.

5

10

9. A task scheduler, according to claim 8, further
comprising:

15

means for determining if any of said plurality of
requests for access is for an isochronous task;
means for assigning a predetermined time period
for each said isochronous task for access to sys-
tem resources; and
means for granting a highest first priority level to
said request for an isochronous task during said
predetermined time period assigned to said task.

20

10. A task scheduler, according to claim 9, wherein
said means for assigning a predetermined time
period for each said isochronous task is program-
mable to permit dynamic modification of said
time period in response to requests for access.

25

30

35

40

45

50

55

7

FIG. 1

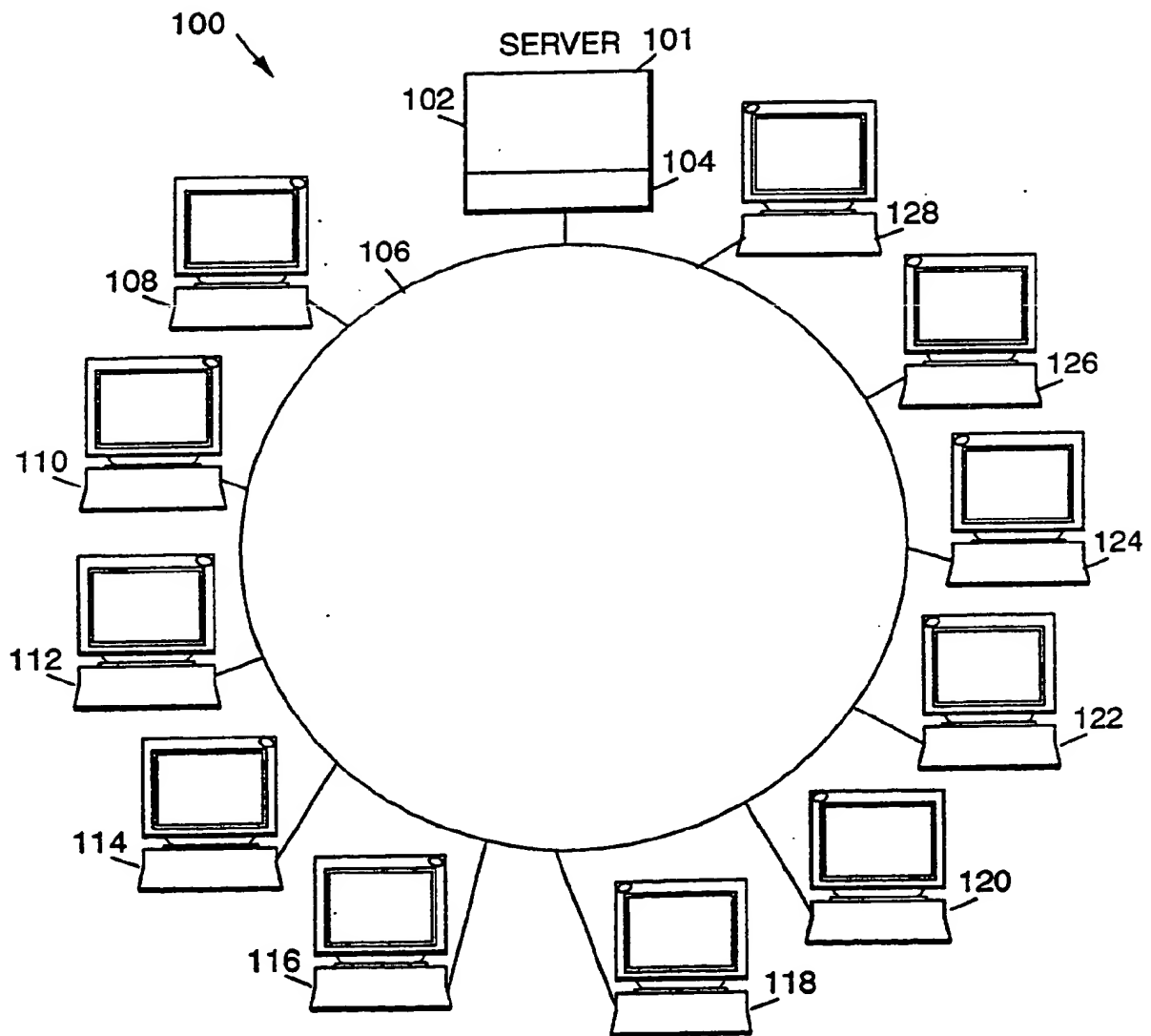


FIG. 2

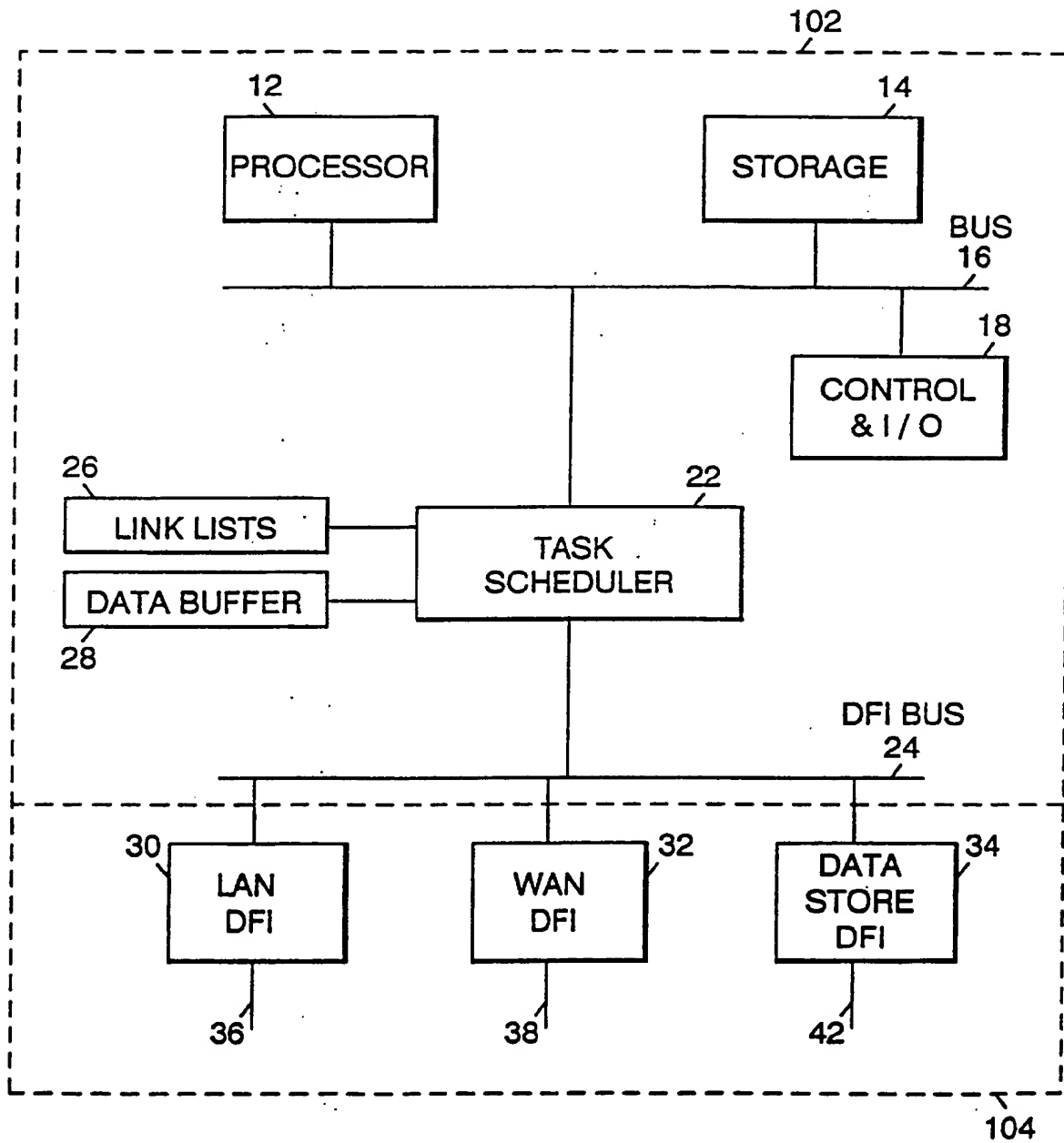


FIG. 3

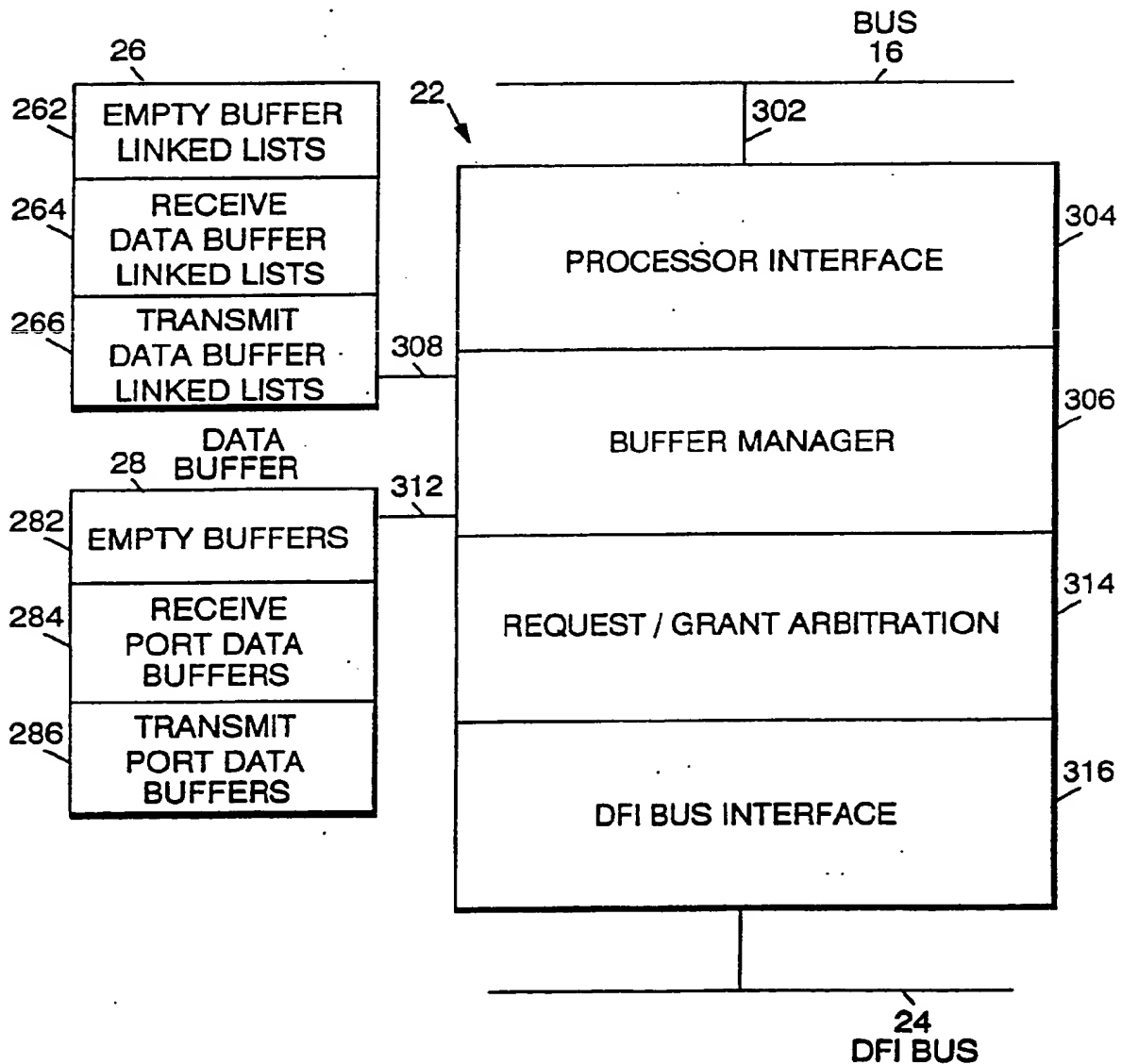


FIG. 4

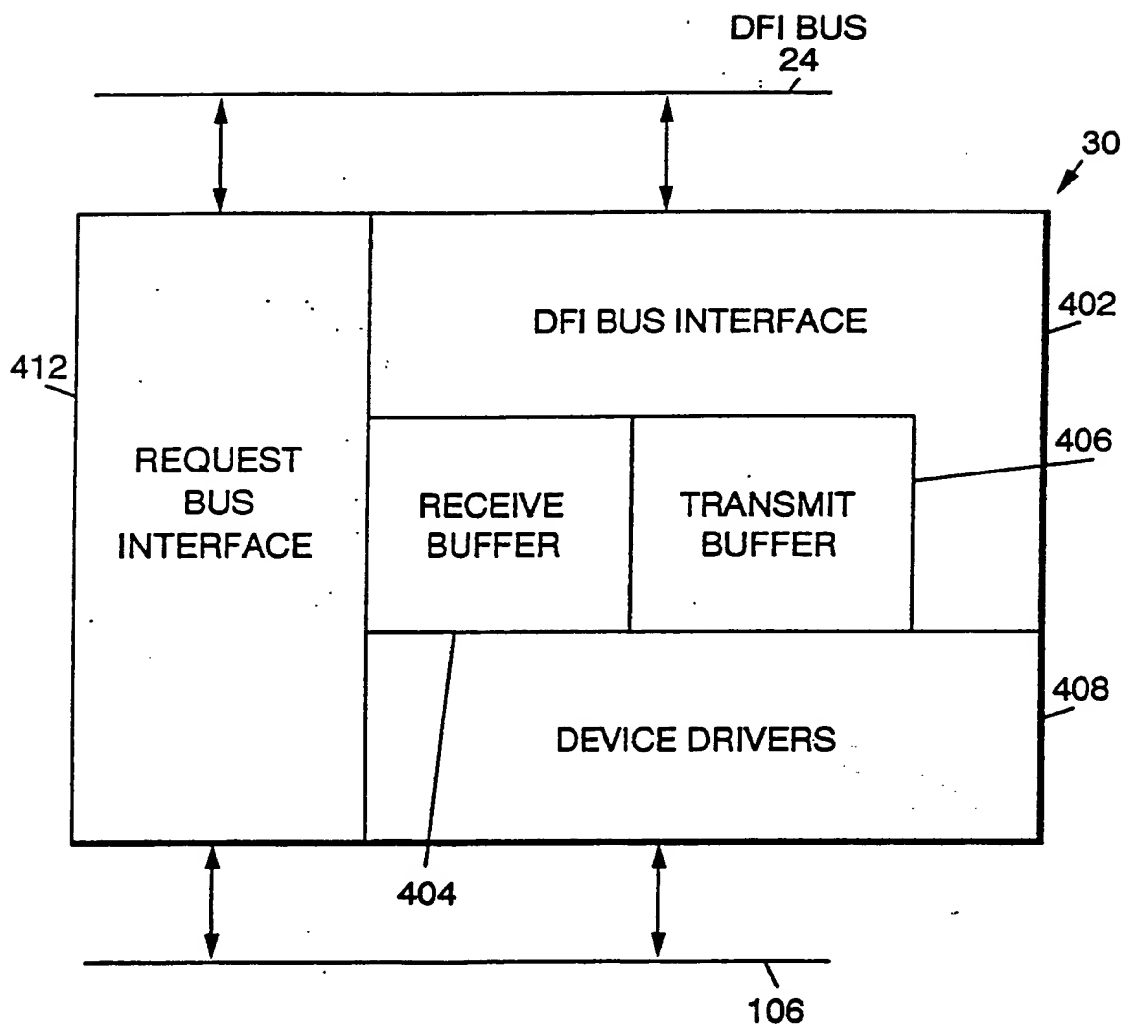


FIG. 5

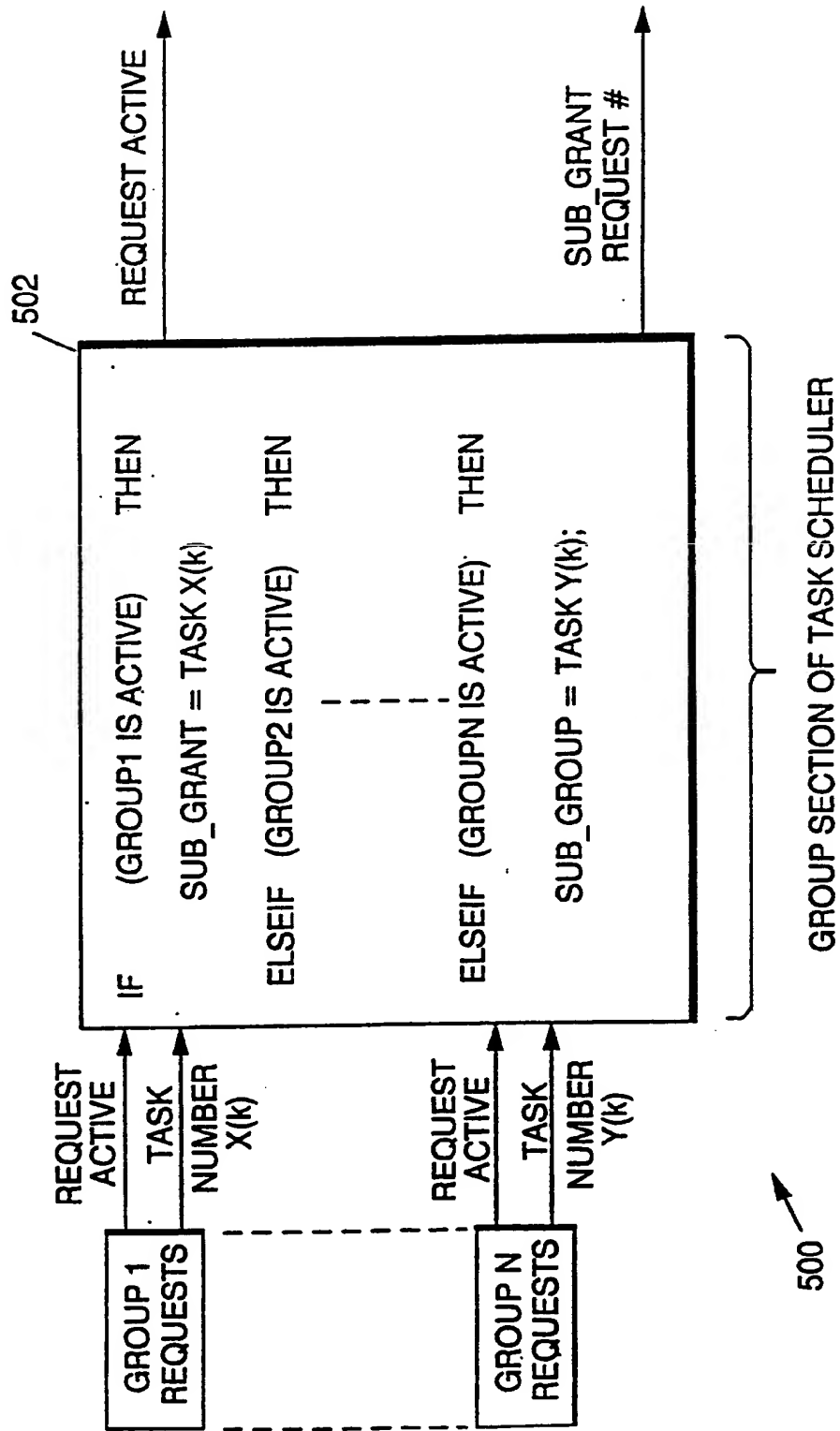


FIG. 6

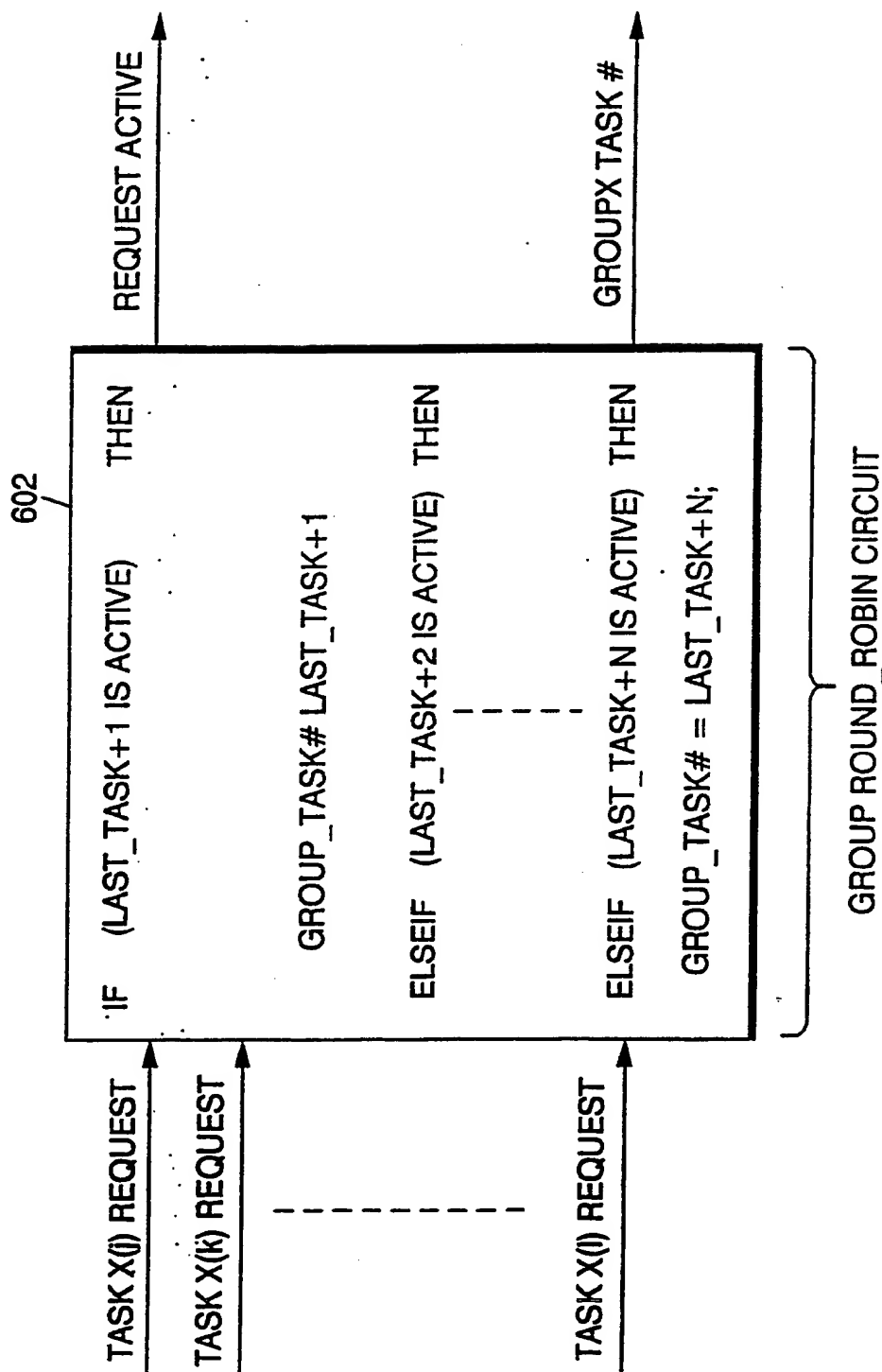


FIG. 7

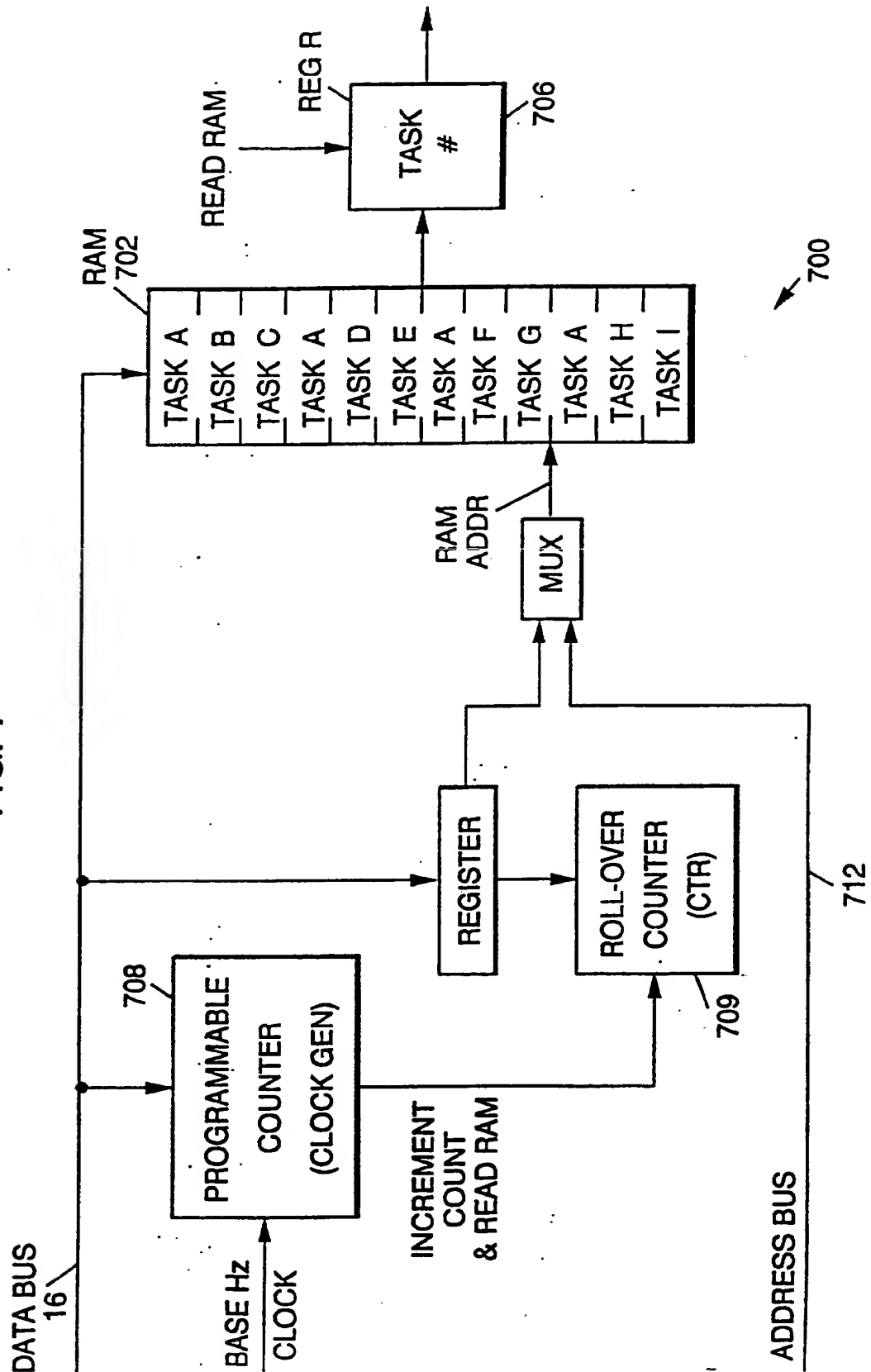


FIG. 8

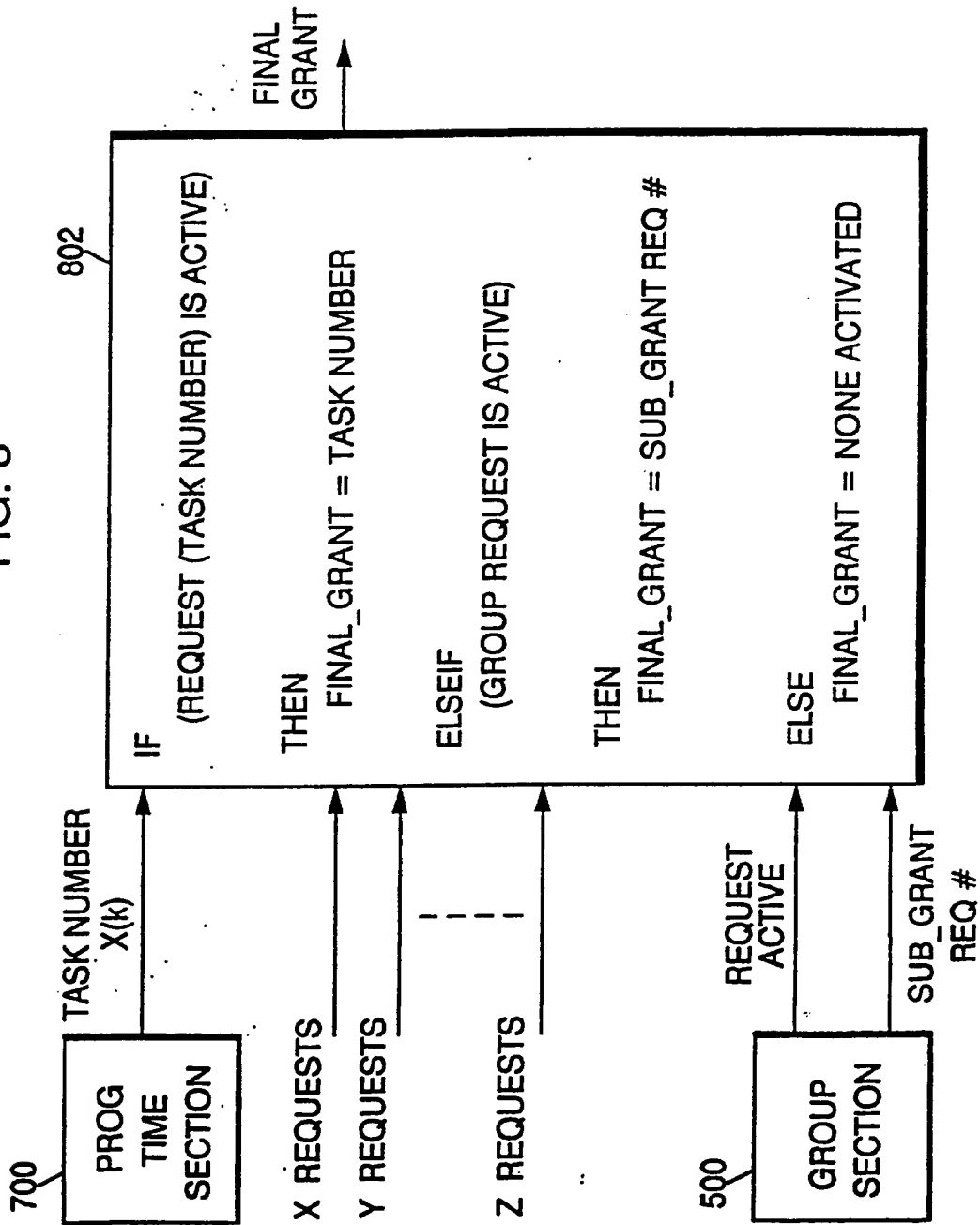


FIG. 9

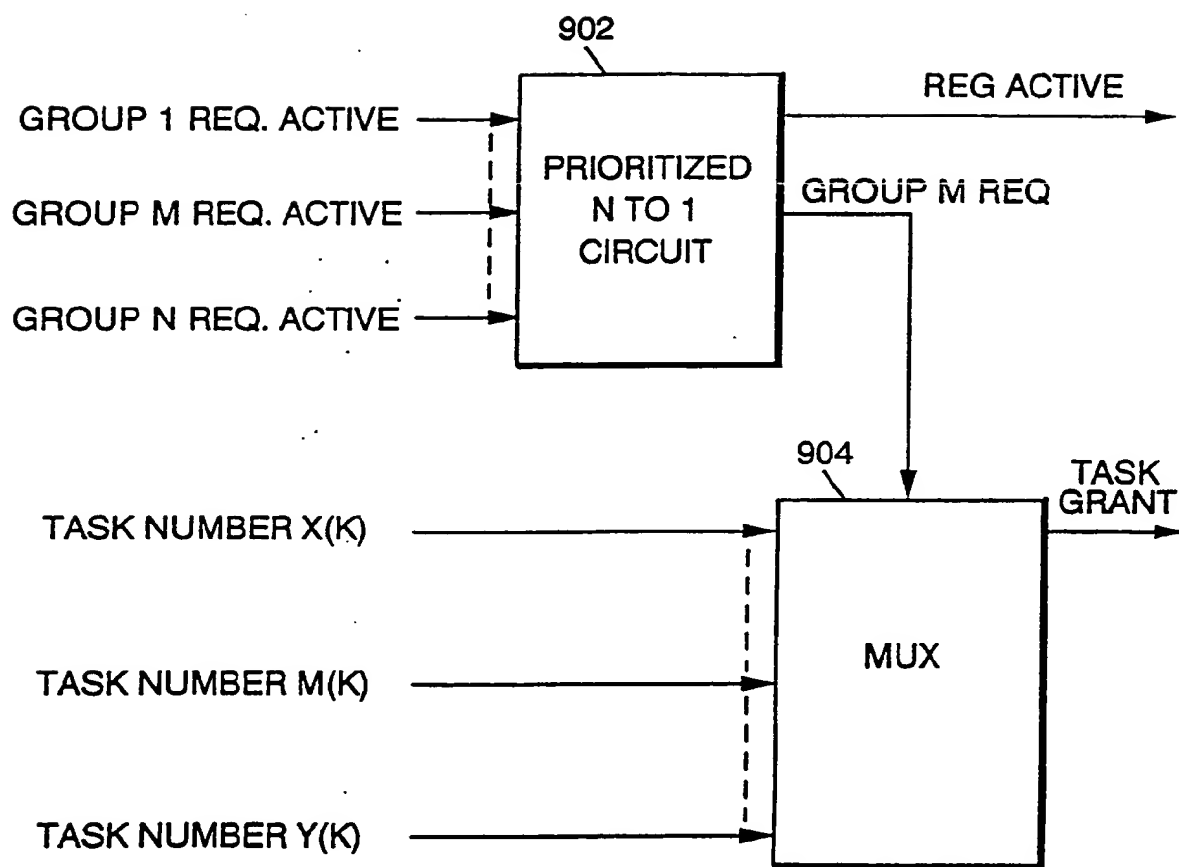


FIG. 10

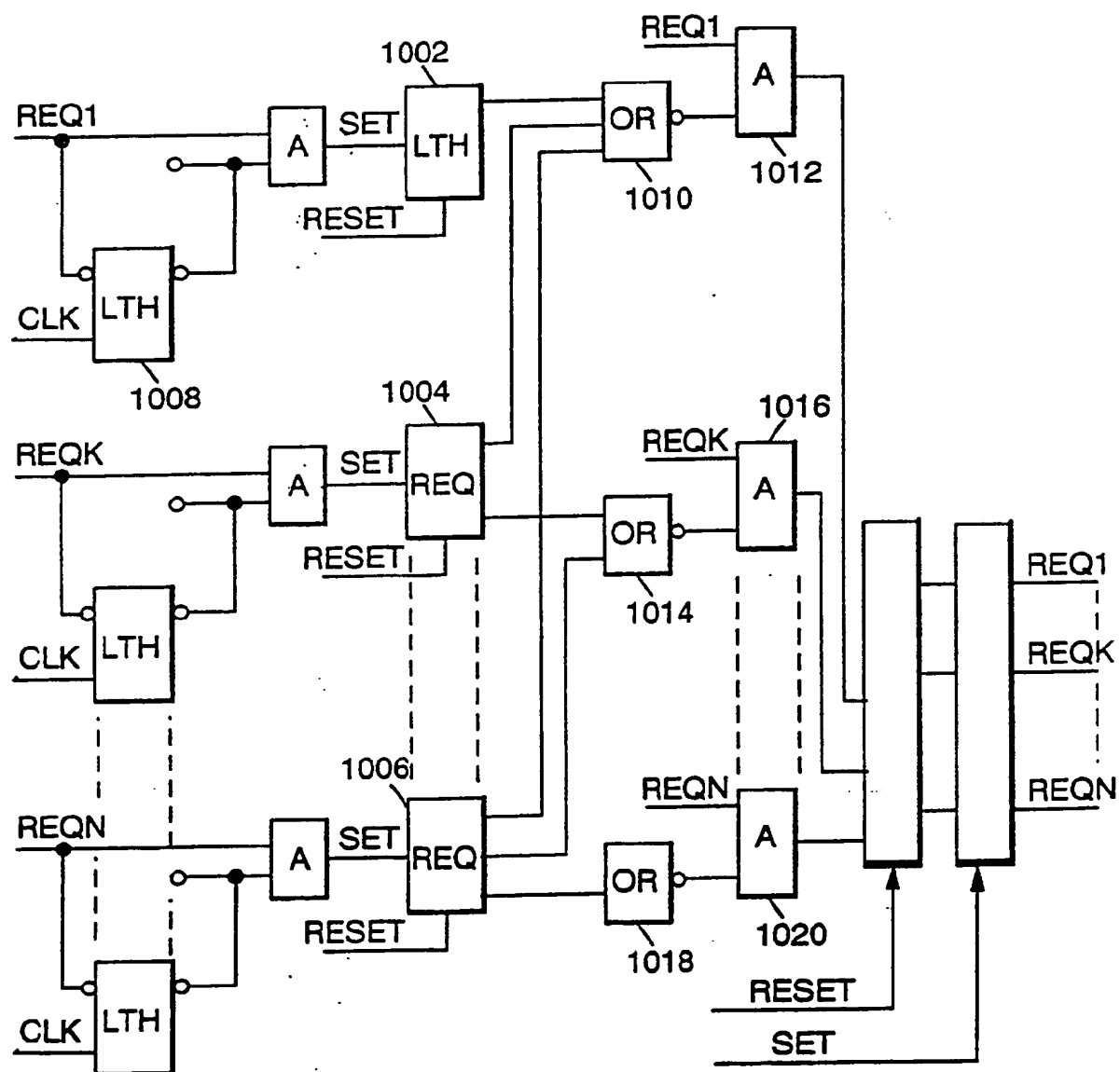
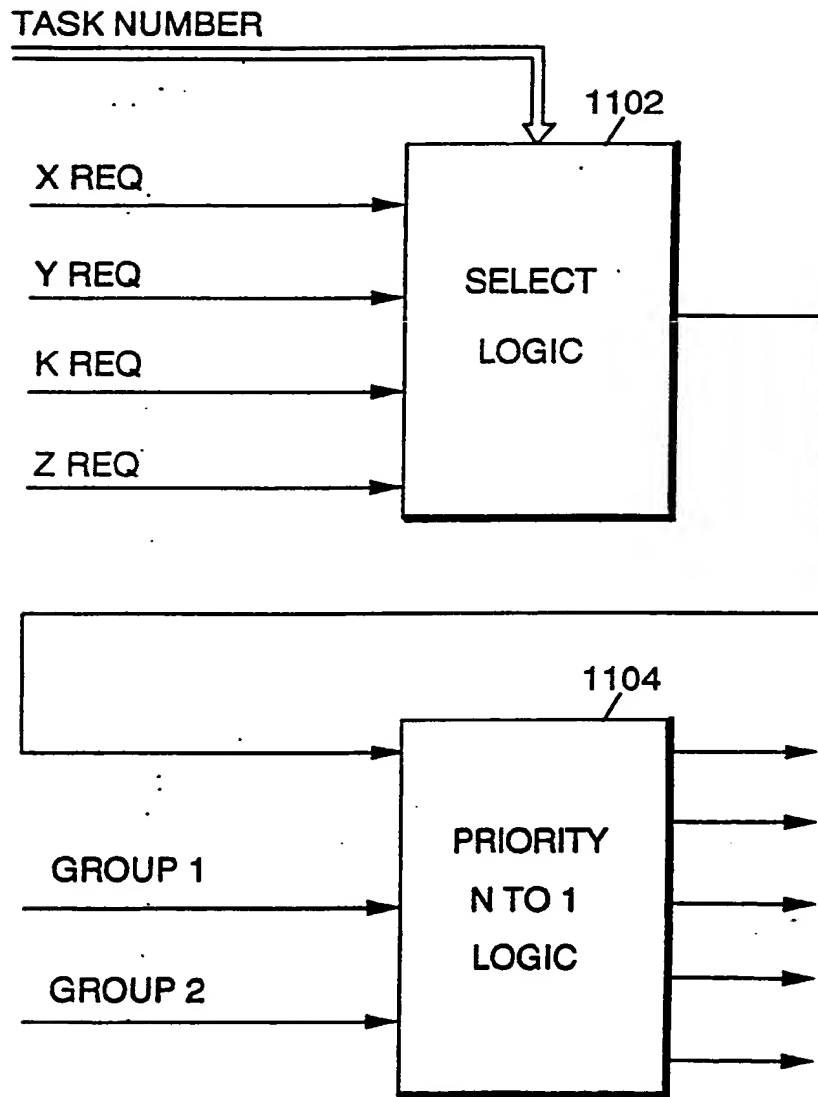


FIG. 11



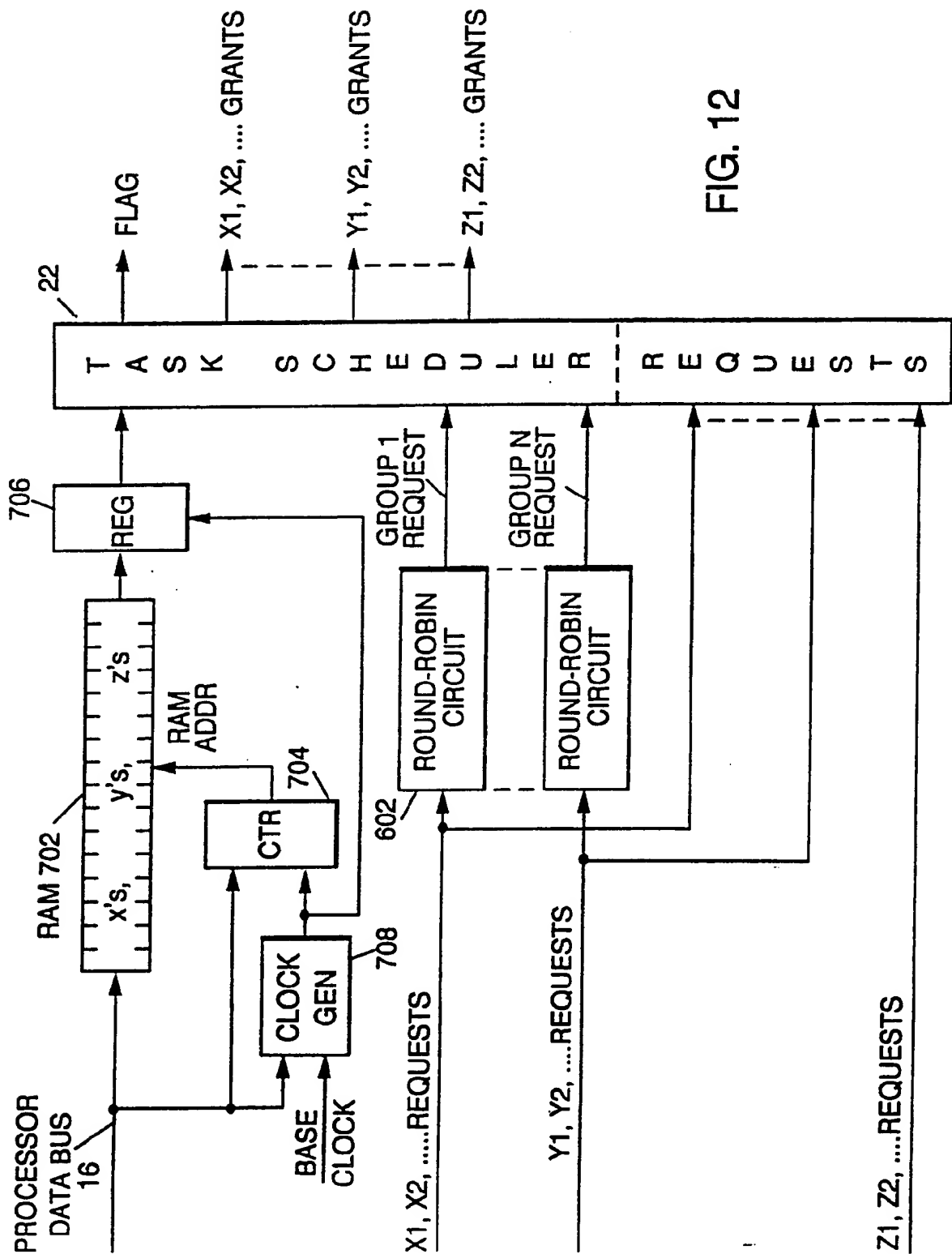


FIG. 12



⑪ Publication number : **0 658 841 A3**

⑫ **EUROPEAN PATENT APPLICATION**

⑲ Application number : **94480106.7**

⑤① Int. Cl.⁶ : **G06F 9/46**

⑳ Date of filing : **26.10.94**

③① Priority : **16.12.93 US 168616**

④③ Date of publication of application :
21.06.95 Bulletin 95/25

⑧④ Designated Contracting States :
DE FR GB

⑧⑧ Date of deferred publication of search report :
19.07.95 Bulletin 95/29

⑦① Applicant : **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

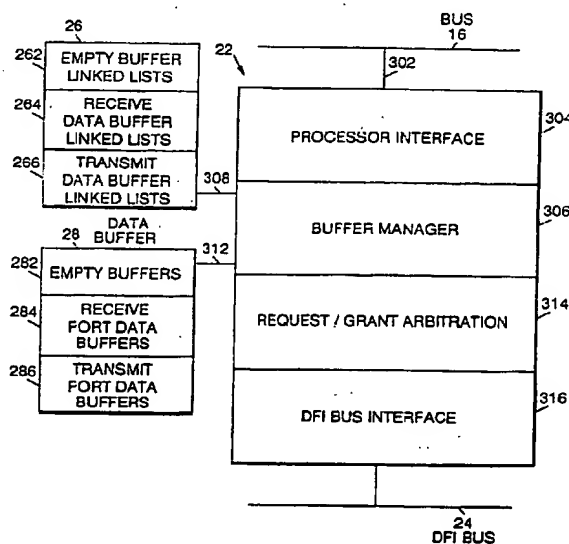
⑦② Inventor : **Bass, Brian Mitchell**
4021 Old Sturbridge Drive
Apex, North Carolina (US)
Inventor : **Ku, Edward Hau-Chun**
115 Lochwood West
Cary, North Carolina (US)
Inventor : **Lin, Bou-Chung**
8712 Fort Macon Court
Raleigh, North Carolina (US)
Inventor : **Sanaye, Simin Hosne**
10301 Roadstead Way
Raleigh, North Carolina (US)

⑦④ Representative : **Lattard, Nicole**
Compagnie IBM France
Département de Propriété Intellectuelle
F-06610 La Gaude (FR)

⑤④ **A data processing system having a dynamic priority task scheduler.**

⑤⑦ A data processing system includes a server which has a task scheduler for receiving task requests for access, and granting access to system resources based upon a multidimensional scheduling technique wherein groups of tasks are assigned a priority level within a priority scheme, and within each group round robin scheduling assures each task will have access to system resources in turn and a dynamically programmable technique for responding to requests from time-dependent isochronous requests for system resources.

FIG. 3



EP 0 658 841 A3



European Patent
Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 48 0106

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	EP-A-0 383 475 (IBM) 22 August 1990 * the whole document *	1,3,5-9	G06F9/46
Y	---	2,4	
Y	MICROPROCESSING AND MICROPROGRAMMING, vol. 28, no. 1 / 05, March 1990 pages 145-150, RAJA D ET AL 'A PORTABLE REAL-TIME KERNEL FOR 8086/80186/80286/80386 BASED SYSTEMS ON IBM-PC' * page 148, left column, line 29 - line 39 *	2,4	
A	---	1-10	
A	EP-A-0 443 782 (AMERICAN TELEPHONE & TELEGRAPH) 28 August 1991 * the whole document *		
A	---	4,6,8	
A	EP-A-0 332 148 (HONEYWELL INC) 13 September 1989 * summary; page 4, lines 16-40 *		
A	---	10	
A	US-A-4 954 948 (HIRA GERALD M ET AL) 4 September 1990 * column 2, lines 19-43 ; column 4, lines 53-68; column 6, lines 38-42 *		
The present search report has been drawn up for all claims			
Place of search THE HAGUE		Date of completion of the search 17 May 1995	Examiner Klocke, L
CATEGORY OF CITED DOCUMENTS X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure F : intermediate document T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document			

EPO FORM 1503 (03.92) (P04C01)